



Kybernetes

Emerald Article: SOFTWARE FEATURES FOR INTELLIGENT INDUSTRIAL ROBOTS

P. CORTI, G. GINI, M. GINI

Article information:

To cite this document: P. CORTI, G. GINI, M. GINI, 1979"SOFTWARE FEATURES FOR INTELLIGENT INDUSTRIAL ROBOTS", Kybernetes, Vol. 8 Iss: 2 pp. 149 - 154

Permanent link to this document:

<http://dx.doi.org/10.1108/eb005517>

Downloaded on: 14-09-2012

To copy this document: permissions@emeraldinsight.com

This document has been downloaded 12 times since 2008. *

Users who downloaded this Article also downloaded: *

Marwa M. Hassan, Stan Gruber, (2008), "Application of discrete-event simulation to study the paving operation of asphalt concrete", Construction Innovation: Information, Process, Management, Vol. 8 Iss: 1 pp. 7 - 22

<http://dx.doi.org/10.1108/14714170810846495>

Fotis Vouzas, (2004), "HR utilization and quality improvement: the reality and the rhetoric - the case of Greek industry", The TQM Magazine, Vol. 16 Iss: 2 pp. 125 - 135

<http://dx.doi.org/10.1108/09544780410523026>

Ancau Mircea, (2012), "Main aspects concerning PCB manufacturing optimization", Circuit World, Vol. 38 Iss: 2 pp. 75 - 82

<http://dx.doi.org/10.1108/03056121211222291>

Access to this document was granted through an Emerald subscription provided by POLITECNICO DI MILANO

For Authors:

If you would like to write for this, or any other Emerald publication, then please use our Emerald for Authors service.

Information about how to choose which publication to write for and submission guidelines are available for all. Please visit www.emeraldinsight.com/authors for more information.

About Emerald www.emeraldinsight.com

With over forty years' experience, Emerald Group Publishing is a leading independent publisher of global research with impact in business, society, public policy and education. In total, Emerald publishes over 275 journals and more than 130 book series, as well as an extensive range of online products and services. Emerald is both COUNTER 3 and TRANSFER compliant. The organization is a partner of the Committee on Publication Ethics (COPE) and also works with Portico and the LOCKSS initiative for digital archive preservation.

*Related content and download information correct at time of download.

SOFTWARE FEATURES FOR INTELLIGENT INDUSTRIAL ROBOTS

P. CORTI, G. GINI and M. GINI

*Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, Piazza Leonardo da Vinci 32,
20133 Milano (Italy)*

(Received February 14, 1978)

The automation of complex handling and assembly operations can be achieved by the introduction of industrial robots, computer-controlled and equipped with simple tactile sensors. A difficult problem in such robots are emergency situations when a defective component part is encountered. The lack of the capability for an automatic emergency recovery is a serious limitation of the present industrial robots. The purpose of this paper is to investigate and illustrate these emergency situations and the problem of emergency recovery, and to propose a solution by suitable software. It is also pointed out that some features of Artificial Intelligence programming languages are suitable for solving this problem in an industrial robot. The problem of suitable software for such robots is then approached on the basis of the obtained results implemented on a UNIVAC 1108 computer, by simulating the behaviour of a SIGMA robot.

1 INTRODUCTION

The problem tackled in this paper is related to the introduction of a reasonable degree of intelligence in industrial robots. This intelligent behaviour is greatly required in order to solve the emergency situations which arise during normal activity.

The robot to which we are referring is a manipulator employed in assembly processes. It can interact with a mini-computer, which has the task of monitoring the execution of the appropriate sequence of elementary actions necessary in order to achieve the assembly of a mechanical system.

This choice is motivated by the interest of the mechanical assembly process. In fact, the automation of such processes is a very developed field in industrial robotics and, since the early studies of intelligent robots,¹⁻⁶ it has been considered of great interest.

Industrial robots in use today are not very easily programmable handling devices that perform simple and repetitive jobs, involving few alternative actions and a minimum of communication with the environment.⁷

Considering the wide range of needs it is, however, possible to single out a vast group of production processes, as the production in small series, or relatively complex processes to be performed on a large number of machines with the frequent manipulation of parts or operations involving the fitting and joining together of parts, for which an easily programmable robot would be desirable.⁸

This paper deals with the emergency situations which happen when a defective component part is encountered in the assembly process.

In fact, about 2% of the mechanical component parts are usually defective and stop the execution of the assembly process. The solution of these events is generally available to the man who has to find out how to recover from the emergency situation in order to start again the deterministic and automatic assembly process.

The purpose of this paper is to investigate such emergency situations and to point out the characteristics of the software able to solve them.⁹

The emergency recovery problem solution is related to the availability of alternative paths from the error point in order to reach again a legal situation. The choice of the new branch to expand and its reentry point in the normal assembly process depends on the error point and on the way in which the emergency problem has been solved.

A non-deterministic programming language seems very apt for these exigencies; consequently, some Artificial Intelligence techniques can be fruitfully employed.¹⁰

In Section 2 the emergency recovery problem is illustrated and in Section 3 a case study is presented; a solution has been implemented on a UNIVAC 1108 computer by simulating the robot behaviour. In Section 4 the implemented solutions are discussed and the most significant aspects of the programming languages employed are pointed out. In Section 5 the relevant aspects of the control

structures and the context mechanism for solving these problems are presented, and the possibility of introducing such features in present industrial robots is discussed.

This work represents an aspect of the research activity, developed at the Politecnico di Milano in collaboration with the Olivetti Company, which is aimed at introducing some new software facilities in the SIGMA robot.

2 THE EMERGENCY SITUATIONS

The industrial robot, to which we are referring, is employed in some mechanical assembly processes in order to build up a mechanical system composed of several parts.

A robot of this class is devoted to the automatic execution of a fixed sequence of elementary assembly operations, which makes up the completely assembled system. Although quite sophisticated as artificial systems, the industrial robots have no ability to overcome sudden difficulties which arise when an emergency situation occurs.

These emergency situations have very different causes,^{4,9,11} but in our analysis we consider only the situations related to the presence of defective component parts. In such an event the execution of one elementary assembly operation fails. The solution of that occurrence is available only to the man, who has to find out how to recover from the emergency situation in order to start again the automatic assembly process.

In fact in the industrial robots there is no plan formation activity,³⁻⁵ but there is only a program constituted by a fixed sequence of steps. The assembly process is then constituted by a linear sequence of applications of elementary operators (deterministic program) as shown in Figure 2.

For the automatic solution of an emergency problem it is necessary to insert some new branches in those points in which the emergency problem arises. The structure of the process is then modified in a graph structure (non-deterministic program). The choice of the branch to extend is not indifferent, but it is related to the desired strategy required to recover from the emergency.

The reentry point of the expanded branch in the deterministic program depends on the point where the emergency has arisen and on the way in which the emergency has been recovered.

This way may be selected as well according to a global strategy, which takes account of additional

information, e.g. the number of tried strategies without success, the cost of the components parts, the cost of the different operations, etc.

3 A CASE STUDY

We now intend to introduce the emergency recovery problem by a case study, which was proposed by the Olivetti Company.

The mechanical subsystem to assembly is a driver of a teletype drum. The driver is composed of four blocks and one bar fixed on the blocks by means of four screws.

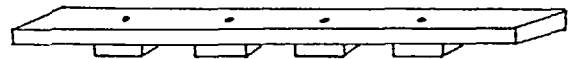


FIGURE 1 Driver of a teletype drum.

The assembly task is performed by a computer-controlled robot SIGMA† which is made up by a mechanical arm, whose hand is used to pick up a component part from its loader, and to put it in the appropriate position of the assembly platform. Moreover the hand is used to screw together one block with the bar.

The elementary operators necessary for executing the assembly activity are as follows:

- TRANSLATE (p,q) defined for the movement of the arm from the position p to the position q;
- GRASP (ob,p) defined for picking up the object ob in the position p;
- UNGRASP (ob,p) defined for leaving the object ob in the position p;
- SCREW (p) defined for the clockwise rotation of a screw in the position p;
- UNSCREW (p) defined for the anticlockwise rotation of a screw in the position p.

In the particular problem these elementary operators are applied in a fixed sequence; therefore it is useful to define two "macro-operators"

- MOVE (ob,p,q,r) which is defined by the sequence "TRANSLATE (p,q); GRASP (ob,q); TRANSLATE (q,r); UNGRASP (ob,r)";


† SIGMA is a programmable industrial robot,⁸ developed and produced by the Ing. C. Olivetti Co. Italy.

DISASSEMBLE (p,q) defined by the sequence "TRANSLATE (p,q); UN-SCREW (q)."

We will utilize:

P1,P2,P3,P4,P5 to indicate the central position of each block and of the bar on the assembly platform;
 L1,L2,L3 to indicate the picking up position from the loaders of the blocks, the bars and the screws;
 W to indicate the wastebin for defective pieces;
 SAVE to indicate a reserved position on the platform;
 POSIN to indicate the initial position of the arm;
 POSFIN to indicate the final position in which the assembled objects are carried out.

The solution of the assembly task is then constituted by the sequence shown in Figure 2. The hand brings an object assembled grasping it by the bar. So it is possible also to write "MOVE(bar,P4,P5, POSFIN)" instead of "MOVE(object,P4,P5, POSFIN)".



```

MOVE (block, POSIN,L1,P1)
MOVE (block, P1,L1,P2)
MOVE (block, P2,L1,P3)
MOVE (block, P3,L1,P4)
MOVE (bar, P4,L2,P5)
MOVE (screw,P5,L3,P1)
SCREW (P1)
MOVE (screw,P1,L3,P2)
SCREW (P2)
MOVE (screw,P2,L3,P3)
SCREW (P3)
MOVE (screw,P3,L3,P4)
SCREW (P4)
MOVE (object,P4,P5,POSFIN)
  
```

FIGURE 2 Sequence of Operations.

The operators which are used to make up the solution of the normal assembly problem do not make use of the DISASSEMBLE macro-operator, which is used only when an emergency arises.

The assembly process previously defined is based on the assumption that all the component parts have no defects, but that is not the real situation. We are now considering only two possible defects of the component parts, because the other may be approached in a very similar way:

i) the holes are defective (in width, pitch or position) in the bar or in some block;

ii) the screws are defective (in pitch, length or width).

Such an emergency forces the robot to interrupt the assembly in the application of the operator SCREW. If we have no information about the reason of the emergency, we can choose among some different strategies, for example:

E1: to change the screw;
 E2: to change the block and the screw on which the assembly process has stopped;
 E3: to unscrew all the screws and to change the bar, the block and the screw on which the assembly has stopped.

Everyone of the illustrated solutions is realized using a suitable sequence of operators; the reentry point in the normal assembly process depends on the strategy.

In this way the assembly process can be represented by a more complex structure, a part of which is shown in Figure 3.

4 SOLUTIONS OF AN EMERGENCY RECOVERY PROBLEM

In the previous Section we have examined an example of mechanical assembly and we have discussed an emergency and its automatic solution.

We shall now illustrate the experimental results which have been obtained on a UNIVAC 1108 computer, using the interpreters of *MICROPLANNER*,¹² and *LISP* and *MAGMALISP*.¹³

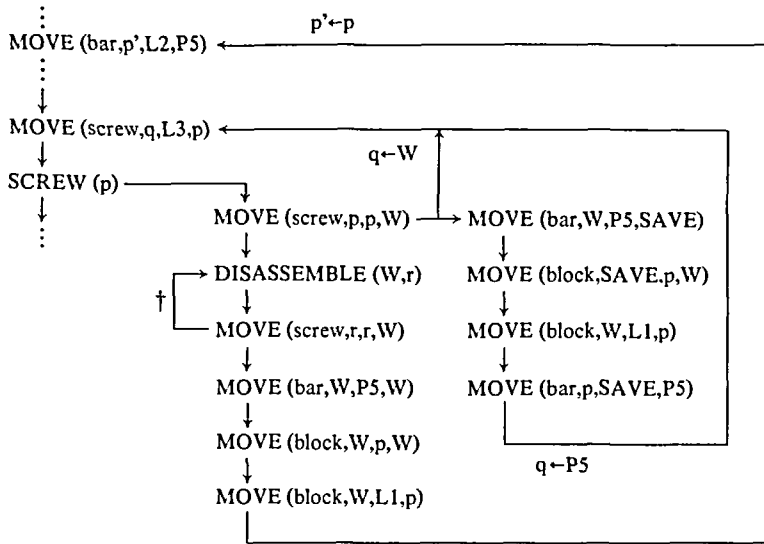
These languages are not available, at the moment, on minicomputers controlling industrial robots, but we intend to investigate which of these classical languages of Artificial Intelligence is the most suitable to solve this kind of problems.

The final aim of this research activity is to design and to implement on a minicomputer an appropriate high-level language for industrial robots.

4.1 MICROPLANNER Solution

The normal assembly process is achieved in *MICROPLANNER* by activating the "ASSEMBLY" theorem.

The emergency problem is simulated by the *LISP* pseudofunction READ and by a casual sequence of T and NIL; T simulates a screw by which it is possible to screw together one block with the bar, NIL simulates a screw which cannot be used owing to some defects.



† the cycle is repeated until there are no screws left.

FIGURE 3 More complex structure for assembly.

When an emergency problem arises, the "EMERGENCY" theorem is activated and each strategy is achieved using three different sequences of goals.

The choice by which a particular strategy is selected is realized by means of flags that are unfurled when a strategy has been tried. This behaviour may be seen as a serious limit for non-determinism, but it is a way to guide the computer in automatically constructing a good solution of the emergency problem; without any information the interpreter of MICROPLANNER would go on changing screws until it finds a screw which can be screwed or until the screws' loader is empty (very expensive and silly situation).

Another very serious problem is how the interpreter carries out the control flow of Figure 3. In fact, all the knowledge about the environment is kept in the database of the external theorem; only this makes it possible to avoid a failure that in an emergency strategy could destroy the results already obtained. When an emergency occurs the control passes to the EMERGENCY theorem, which realizes the different strategies E1, E2 and E3.

When a sequence of emergency is ended the control returns to the ASSEMBLY theorem through EMERGENCY one. In this last theorem are stored all the data that the program requires to end an emergency strategy or to choose a new strategy after the failure of the last tried. In the database of the

ASSEMBLY theorem are stored the following: the state of the assembly process, the point in which it has stopped and from which the deterministic program has to go on.

In Figure 4 we represent the flow of control realized by a MICROPLANNER interpreter.

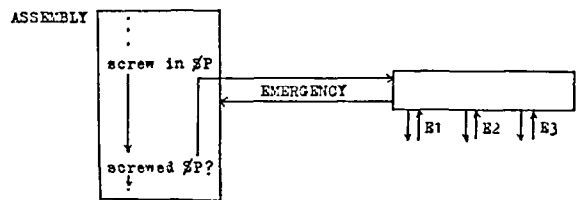


FIGURE 4 Control by MICROPLANNER.

4.2 LISP Solution

The philosophy for the solution of the problem using LISP is quite different. We define one function for every operator introduced in Section 3 with suitable parameters. It is also possible to make a different use of the parameters, because in LISP there are several possibilities for defining and evaluating global and local variables.

In LISP the emergency recovery has to be interpreted as an operator or a set of different operators which are called when there is a fault, and which produces the recovery from the emergency and returns the control to the normal assembly process.

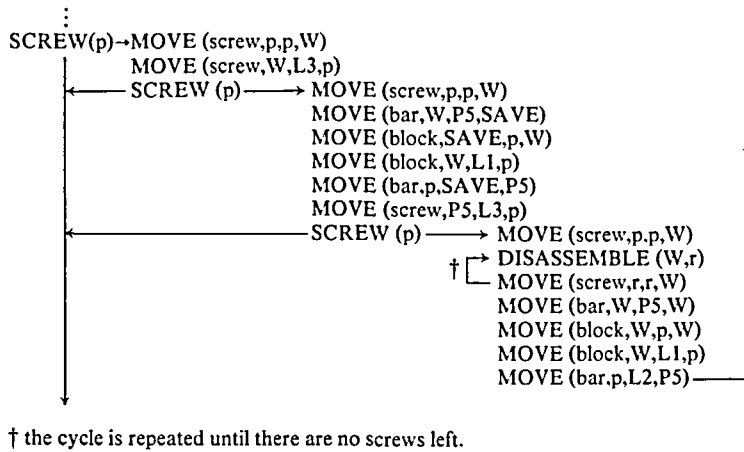


FIGURE 5 LISP solution.

In order to use different strategies it is necessary to have a control structure like that realized by the ATTEMPT function.†

With this function we can obtain a structure very similar to the previous one, or also a structure like that one shown in Figure 5.

As in the MICROPLANNER program the result of the evaluation is the sequence of the elementary actions of the robot necessary to solve the particular assembly problem.

4.3 MAGMALISP Solution

MAGMALISP¹³ is an extended LISP system, proposed by the authors as an implementation tool for Artificial Intelligence languages. The fundamental idea of MAGMALISP is that a tree structure of independent computation environments (context tree) is the supporting structure of any non-deterministic system. The Bobrow and Wegbreit's¹⁴ model of control structures, which allows the user to define his own heuristics, is on the basis of this language.

The main idea of MAGMALISP is the notion of context, as a complete environment capable of deterministic computations, which includes also the control structure. Non-determinism, necessary for solving emergency recovery problem, is attained by exploring different alternatives in different contexts or by generating, deleting and switching among different procedures.

† The ATTEMPT function allows a particular evaluation of an expression; if an error of a defined type arises, the system stops the evaluation and goes to another expression which allows a recovery from the error situation.

The system acts like a system with many independent computation environments in which it is possible to transfer information from one context to another.

According to these features this language seems to be a good solution of the emergency recovery problem, not only in the proposed case, but also in complex situations. In fact, using MAGMALISP it is possible to break off the evaluation in the normal sequence when an emergency arises, to execute a chosen emergency strategy and to reenter in the right point.

In the proposed case the control structure is similar to that one of Figure 5, but the control flow is programmed in a very natural way, because for each procedure an access environment, a return point and a context would be explicitly expressed.

5 CONCLUDING REMARKS

We have proposed a solution of the emergency recovery problem by examining a simple case of mechanical assembly process on which the most relevant aspects of the emergency situations can be focused. We have examined the possibility of solving the problem by particular features of some Artificial Intelligence programming languages. In order to perform our task, which is the introduction of some emergency recovery capabilities in industrial robots, we have now to draw some conclusions.

A language with some capabilities to deal with non-deterministic programs is desirable. Since the most important part of the assembly is performed by a fixed sequence of steps, the non-determinism is

mainly required in conjunction with a flexible control structure, which can easily manage the activation and the interruption of the processes.

On the other hand, the usual non-deterministic languages can not be practicable for a mini-computer connected with the industrial robot.

It is then crucial to find the only important features to be introduced into new software for robots. The result outlined in this paper constitutes the basis for the implementation project of an extended *LISP* system (like *MAGMALISP*) in a *LABEN* minicomputer connected with our *SIGMA* robot.

REFERENCES

1. T. Winograd, "Procedures as a representation for data in a computer program for understanding natural languages" *MAC TR-84* (M.I.T., Cambridge, Mass., 1971).
2. R. E. Fikes and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving" *Proc. 2nd IJCAI, London, England (1971)*.
3. S. E. Fahlman, "A planning system for robot construction tasks" *Artificial Intelligence 5* (1974).
4. E. D. Sacerdoti, "A structure for plans and behaviour", A.I. Center, Tech. N. 109 (S.R.I., Menlo Park, California, 1975).
5. A. Tate, "Interacting goals and their use" *Proc. 4th IJCAI Tbilisi, Georgia (1975)*.
6. R. Waldinger, "Achieving several goals simultaneously" A.I. Center, Tech. N. 107 (S.R.I., Menlo Park, California, 1975).
7. R. G. Abraham, J. F. Beres and M. Yaroshuk, "Requirements analysis and justification of intelligent robots" *Proc. 5th ISIR, Chicago, Illinois (1975)*.
8. A. D'Auria, and M. Salmon, "SIGMA: an integrated general purpose system for automatic manipulation" *Proc. 5th ISIR, Chicago, Illinois (1975)*.
9. G. Gini, M. Gini and M. Somalvico, "Emergency recovery in intelligent robots" *Proc. 5th ISIR, Chicago, Illinois (1975)*.
10. D. Bobrow and B. Raphael, "New programming languages for Artificial Intelligence" *Computing Surveys 6, 3* (1974).
11. P. H. Winston, "Water's arm dynamics theory" in *New progress in Artificial Intelligence AI-TR-310* (M.I.T., Cambridge, Massachusetts, 1974).
12. G. J. Sussman, T. Winograd and E. Charniak, "MICRO-PLANNER Reference manual" *AI TR-203* (M.I.T., Cambridge, Mass. 1970).
13. C. Montangero, G. Pacini and F. Turini, "MAGMALISP: a machine language for Artificial Intelligence" *Proc. 4th IJCAI, Tbilisi, Georgia (1975)*.
14. D. G. Bobrow and B. Wegbreit, "A model for control structures for Artificial Intelligence programming languages" *Proc. 3rd IJCAI, Stanford, California (1973)*.